

---

## Investigación Operativa

---

### Finding magic squares with Douglas--Rachford algorithm

**Francisco J. Aragón Artacho**

Departamento de Matemáticas  
Universidad de Alicante  
✉ francisco.aragon@ua.es

**Paula Segura Martínez**

Departamento de Estadística e Investigación Operativa  
Universidad de Valencia  
✉ paulamaths@gmail.com

#### Abstract

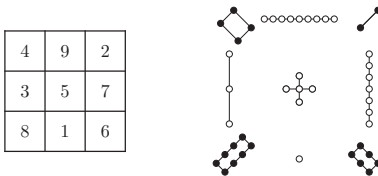
In this expository paper, we show how to use the Douglas--Rachford algorithm as a successful heuristic for finding magic squares. The Douglas--Rachford algorithm is an iterative projection method for solving feasibility problems. Although its convergence is only guaranteed in the convex setting, the algorithm has been successfully applied to a number of similar nonconvex problems, such as solving Sudoku puzzles. We present two formulations of the nonconvex feasibility problem of finding magic squares, which are inspired by those of Sudoku, and test the Douglas--Rachford algorithm on them.

**Keywords:** Douglas--Rachford algorithm, magic square, feasibility problem, nonconvex constraints.

**AMS Subject classifications:** 47J25, 47N10, 05B15, 97A20.

## 1. Introduction

A *magic square* of order  $n$  is an  $n \times n$  matrix whose elements are the positive integers  $1, 2, \dots, n^2$  arranged in such a way that the sum of the  $n$  numbers in any horizontal, vertical, or main diagonal line is always the same number, known as the *magic constant*. Clearly, the magic constant is equal to  $n(n^2 + 1)/2$ , as it is the result of dividing by  $n$  the sum of  $1, 2, \dots, n^2$ . Hence, for magic squares of order 3, 4 and 5, their respective magic constants are 15, 34 and 65. In Figure 1 we show the unique magic square of order 3, except for rotations and reflections.

Figure 1: Modern and traditional representations of the *luoshu* magic square

Magic squares have a long history and were frequently associated with mystical and supernatural properties. It is believed that the magic square of order 3 shown in Figure 1, commonly referred by its Chinese name *luoshu*, was known to Chinese mathematicians long before 500 BCE. According to [26], the very first textual reference to the *luoshu* appears to be in the writings of Zhuang Zi (369–286 B.C.E.), one of the founders of Daoism. The name *luoshu* means “Luo River writing”, and refers to the following legend:

*There was a huge flood in the ancient China. While Sage King Yu stood on the banks of the Luo River trying to channel the water out to the sea, a turtle with a curious pattern of dots arranged on its shell emerged from the river. This pattern was the magic square of order 3. Thereafter people were able to use this pattern in a certain way to control the river and protect themselves from floods.*

It is possible to construct magic squares of any order, except for  $n = 2$ : if the following grid was a magic square,

a	b
c	d

then one should have  $a + b = a + c$ , which implies  $b = c$ .

Probably, the most well-known magic square of order 4 is the one immortalized by the German artist Albrecht Dürer in his engraving *Melencolia I*, see the detail in Figure 2. Observe how the date of the engraving, 1514, is shown in the two middle cells of the bottom row. This magic square has the additional property that the sums in any of the four quadrants, as well as the sum of the middle four numbers, are all 34 (which is called a *gnomon magic square*).

Dürer became very interested in mathematics and devoted his life to the study of linear perspective and proportion. He claimed:

*The new art must be based upon science – in particular, upon mathematics, as the most exact, logical, and graphically constructive of the sciences.*



Figure 2: Albrecht Dürer, Melencolia I (detail)

Dürer's magic square previously appeared in the manuscript *De viribus quantitatis* by the Franciscan friar Luca Pacioli, which includes a short chapter on magic squares dating from about 1501–1503. In that chapter, Pacioli includes one example each of magic square of orders 3 to 9, and the one of order 4 is identical to the one in *Melancolia I*. Note that the chances for coincidence are very low, as there are 880 magic squares of order 4 (multiplied by their rotations and reflections). It is very plausible that Dürer learnt this magic square from Pacioli, during his second visit to Italy between 1505–1507. For more details, see [26, 27].

There are many ways of constructing magic squares. In this work, based on [24], we present a novel approach by formulating their construction as a nonconvex feasibility problem which can be tackled by the Douglas–Rachford algorithm. Of course, this is only an expository problem, as there are much more efficient strategies for constructing magic squares. We review the main characteristics of this algorithm in Section 2. Two different formulations are presented in Section 3. The numerical experiments reported Section 4 demonstrate that

the Douglas--Rachford algorithm can be used as a heuristic for finding them. We finish with some conclusions in Section 5.

## 2. Projection algorithms

Given a finite family  $C_1, C_2, \dots, C_N$  of subsets of a Hilbert space  $\mathcal{H}$ , a *feasibility problem* consists in finding a point in their intersection; that is,

$$\text{Find } x \in \bigcap_{i=1}^N C_i. \quad (2.1)$$

Thanks to the generality of (2.1), plenty of problems can be modeled as feasibility problems. In many cases, finding a point in the intersection of the sets is a challenging task, but computing the projection of points onto each of the sets is easy. In such scenarios, *projection methods* become very useful. These are algorithms that use the projection of points onto the sets to define the iterates. Recall that the *projection* of a point  $x$  onto a closed set  $C \subset \mathcal{H}$  is defined by

$$P_C(x) := \left\{ z \in C : \|x - z\| = \inf_{c \in C} \|x - c\| \right\},$$

where  $\|\cdot\|$  denotes the norm induced by the inner product  $\langle \cdot, \cdot \rangle$  of  $\mathcal{H}$ . Observe that, in general, the projection operator  $P_C : \mathcal{H} \rightrightarrows C$  is a set-valued mapping, see Figure 3. Nonetheless, if the set  $C$  is nonempty, closed and convex, the projection is uniquely determined as follows (see, e.g., [8, Theorem 3.16]):

$$p = P_C(x) \iff p \in C \text{ and } \langle c - p, x - p \rangle \leq 0, \forall c \in C. \quad (2.2)$$

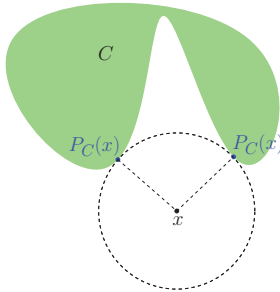


Figure 3: Projection of a point  $x$  onto a nonconvex set  $C$

The most intuitive and well-known projection method is the von Neumann's

alternating projection algorithm, which iterates by cyclically projecting onto each of the sets, see Figure 4.

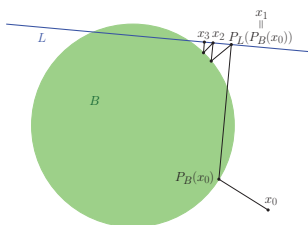


Figure 4: Alternating projection method applied to a ball and a line in  $\mathbb{R}^2$

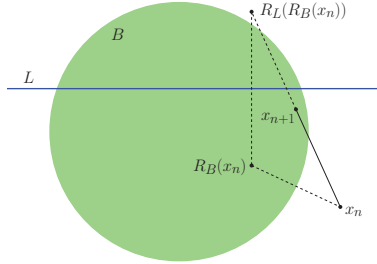
In this work, we will focus on another popular projection method: the *Douglas--Rachford algorithm*. Although the scheme has its origins in the work of J. Douglas and H.H. Rachford [14], where it was proposed for solving a system of linear equations arising in heat conduction problems, it was P.L. Lions and B. Mercier [19] who really deserve the credit for the algorithm. In their work, not only they showed how to successfully generalize the algorithm for solving convex feasibility problems, but they actually provided a splitting algorithm for finding a zero in the sum of two maximally monotone operators.

The Douglas--Rachford algorithm employs the reflection operators when is applied for solving feasibility problems. Recall that the *reflection* of a point  $x \in \mathcal{H}$  onto a closed set  $C$  is defined as  $R_C(x) := 2P_C(x) - x$ . The iterates of the algorithm for solving feasibility problems involving two sets are obtained by computing an average between the current point and two consecutive reflections onto the sets, see Figure 5. For this reason, the scheme is also referred as the *averaged alternating reflections method*.

The Douglas--Rachford algorithm has recently gained great attention, in part thanks to its good performance as a heuristic in nonconvex settings (where the convergence of the algorithm is not guaranteed). In the remainder of this section we will review its main properties in the convex setting, we will show how the algorithm can be applied to any number of sets thanks to a product space reformulation of the feasibility problem, and we will review an application in the nonconvex setting for solving Sudoku puzzles.

## 2.1. The Douglas--Rachford algorithm for convex sets

When the Douglas--Rachford algorithm is applied to two closed and convex sets  $A, B \subset \mathcal{H}$ , it can be viewed as a fixed point iterative method defined by the *Douglas--Rachford operator*  $T_{A,B} := \frac{1}{2}(\text{Id} + R_B R_A)$ , where  $\text{Id}$  denotes the

Figure 5: Douglas--Rachford method applied to a ball and a line in  $\mathbb{R}^2$ 

identity mapping:

$$x_{n+1} = T_{A,B}(x_n) \quad \text{for } n = 0, 1, 2, \dots$$

Note that the iterates are uniquely determined because the reflection operators onto convex sets are single-valued, as the projection operators are. Further, observe that  $\bar{x} \in \mathcal{H}$  is a fixed point of the operator  $T_{A,B}$  if and only if

$$\bar{x} = 2P_B(2P_A(\bar{x}) - \bar{x}) - (2P_A(\bar{x}) + \bar{x}) \iff P_A(\bar{x}) = P_B(2P_A(\bar{x}) - \bar{x}),$$

which implies that  $P_A(\bar{x}) \in A \cap B$ . Therefore, the projection onto the set  $A$  of any fixed point of the Douglas--Rachford operator solves the feasibility problem.

Using a well-known theorem of Opial [20, Theorem 1], the sequence generated by the algorithm can be proved to be weakly convergent to a fixed point of the Douglas--Rachford operator, whenever  $A \cap B \neq \emptyset$ . This is a consequence of the *firm nonexpansivity* of the projection operator, which means

$$\|P_A(x) - P_A(y)\|^2 + \|(\text{Id} - P_A)(x) - (\text{Id} - P_A)(y)\|^2 \leq \|x - y\|^2, \quad \forall x, y \in \mathcal{H}.$$

The latter implies that the reflection operator is *nonexpansive* (or Lipschitz continuous with constant 1), i.e.,

$$\|R_A(x) - R_A(y)\| \leq \|x - y\|, \quad \forall x, y \in \mathcal{H},$$

and therefore,  $T_{A,B}$  is firmly nonexpansive [17, Theorem 12.1].

The next result summarizes the main well-known properties of the Douglas--Rachford algorithm in the convex setting.

Let  $A, B \in \mathcal{H}$  be closed and convex sets. Given any  $x_0 \in \mathcal{H}$ , define  $x_{n+1} = T_{A,B}(x_n)$ , for every  $n \geq 0$ . Then, the following holds:

- (i) If  $A \cap B \neq \emptyset$ , then  $\{x_n\}$  is weakly convergent to a point  $x^*$  and  $\{P_A(x_n)\}$  is weakly convergent to  $P_A(x^*) \in A \cap B$ ,
- (ii) If  $A \cap B = \emptyset$ , then  $\|x_n\| \rightarrow \infty$ .

(i) For the first part, see [9, Theorem 3.13 and Corollary 3.9]. Note that the weak convergence of the *shadow sequence*  $\{P_A(x_n)\}$  cannot be derived from the weak convergence of  $\{x_n\}$ , since  $P_A$  may not be weakly continuous (see [8, Example 4.20]). The weak convergence of the shadow sequence was proved by Svaiter in [25, Theorem 1]. (ii) See [7, Corollary 2.2]. ■

In contrast with the alternating projection algorithm, whose scheme can be easily generalized to more than two sets by cyclically projecting onto the sets, the same cannot be done with the Douglas--Rachford algorithm (see [1, Example 2.1] for a simple example involving three lines in  $\mathbb{R}^2$ ). Fortunately, the product space formulation, due to Pierra [22], permits to reformulate any feasibility problem as another one involving only two sets. This is the subject of the next subsection.

## 2.2. Product space formulation

Consider the feasibility problem 2.1 given by  $N$  sets. Let

$$\mathcal{H}^N := \mathcal{H} \times \cdots \times \mathcal{H} = \{\mathbf{x} = (x_1, \dots, x_N) : x_i \in \mathcal{H}, i = 1, \dots, N\}.$$

It can be easily checked that the space  $\mathcal{H}^N$  endowed with the inner product  $\langle \mathbf{x}, \mathbf{y} \rangle := \sum_{i=1}^N \langle x_i, y_i \rangle$  is a Hilbert space. Let us consider two sets  $C, D \subset \mathcal{H}^N$  defined by

$$C := C_1 \times C_2 \times \cdots \times C_N \quad \text{and} \quad D := \{(x, x, \dots, x) \in \mathcal{H}^N : x \in \mathcal{H}\}.$$

The set  $D$ , which is sometimes called the *diagonal*, is always a closed subspace. Observe that the feasibility problem 2.1 is equivalent to the one involving the sets  $C$  and  $D$ , since

$$x \in \bigcap_{i=1}^N C_i \iff (x, x, \dots, x) \in C \cap D.$$

Not only 2.1 can be recast as a feasibility problem involving two sets, but also the projection onto each of these sets has a closed form in terms of  $P_{C_1}, P_{C_2}, \dots, P_{C_N}$ , as shown next. For completeness, we include a proof based on [2, Proposition 3.1]. This result extends Pierra's Lemma 1.1 in [22] to possibly nonconvex sets.

For any  $\mathbf{x} = (x_1, x_2, \dots, x_N) \in \mathcal{H}^N$ , one has

$$P_D(\mathbf{x}) = \left( \frac{1}{N} \sum_{i=1}^N x_i, \dots, \frac{1}{N} \sum_{i=1}^N x_i \right),$$

and if  $P_{C_1}(x_1), P_{C_2}(x_2), \dots, P_{C_N}(x_N)$  are nonempty,

$$P_C(\mathbf{x}) = P_{C_1}(x_1) \times P_{C_2}(x_2) \times \dots \times P_{C_N}(x_N).$$

Let  $\mathbf{p} = (p, \dots, p) \in D$  be the projection of  $\mathbf{x}$  onto the set  $D$ . For any  $d \in \mathcal{H}$ , one has  $\mathbf{d} := (d, \dots, d) \in D$ . Then, using the characterization 2.2 applied to  $\mathbf{p} + \mathbf{d}$  and  $\mathbf{p} - \mathbf{d}$ , we deduce

$$0 = \langle \mathbf{d}, \mathbf{x} - \mathbf{p} \rangle = \sum_{i=1}^N \langle d, x_i - p \rangle = \left\langle d, \sum_{i=1}^N x_i - Np \right\rangle.$$

As the latter holds for any  $d \in \mathcal{H}$ , it must be  $p = \frac{1}{N} \sum_{i=1}^N x_i$ , as claimed.

To prove the formula for the projection onto the set  $C$ , pick first any point  $\mathbf{p} = (p_1, \dots, p_N) \in P_{C_1}(x_1) \times \dots \times P_{C_N}(x_N) \subseteq C$ . Then, for any  $\mathbf{c} = (c_1, \dots, c_N) \in C$ , one has

$$\|\mathbf{x} - \mathbf{c}\|^2 = \sum_{i=1}^N \|x_i - c_i\|^2 \geq \sum_{i=1}^N \|x_i - p_i\|^2 = \|\mathbf{x} - \mathbf{p}\|^2.$$

Therefore,  $P_C(\mathbf{x}) \supseteq \prod_{i=1}^N P_{C_i}(x_i)$ .

Conversely, let  $\mathbf{p} = (p_1, \dots, p_N) \in P_C(\mathbf{x})$ . Pick any  $j \in \{1, \dots, N\}$  and choose any  $c_j \in C_j$ . Let  $q_i := p_i$  if  $i \neq j$  and let  $q_j := c_j$ , and define  $\mathbf{q} := (q_1, \dots, q_N) \in C$ . Then,

$$\sum_{i=1}^N \|x_i - p_i\|^2 = \|\mathbf{x} - \mathbf{p}\|^2 \leq \|\mathbf{x} - \mathbf{q}\|^2 = \sum_{i=1}^N \|x_i - q_i\|^2 = \sum_{i=1, i \neq j}^N \|x_i - p_i\|^2 + \|x_j - c_j\|^2,$$

whence,  $\|x_j - p_j\| \leq \|x_j - c_j\|$ . Since  $c_j$  was arbitrarily chosen in  $C_j$ , this implies that  $p_j \in P_{C_j}(x_j)$ , and concludes the proof. ■

Pierra's product space formulation works well in practice whenever the number of sets  $N$  is small. When  $N$  is large, the computational cost of calculating the iterates in the large space  $\mathcal{H}^N$  instead of  $\mathcal{H}$  becomes prohibitive. In these cases, the cyclic Douglas--Rachford algorithm with  $r$ -sets-Douglas--Rachford operators, recently introduced in [6], would be a better choice. The latter is a generalization of the cyclic Douglas--Rachford method of Borwein--Tam [13], whose iterates are defined by

$$x_{n+1} = T_{C_N, C_1} T_{C_{N-1}, C_N} \dots T_{C_2, C_3} T_{C_1, C_2}(x_n);$$



that is, a cyclic composition of Douglas–Rachford operators applied to the sets.

### 2.3. The Douglas–Rachford algorithm for nonconvex sets

Theorem 2.1 only guarantees the global convergence of the Douglas–Rachford algorithm for convex sets. In spite of this, the algorithm has been successfully applied as a heuristic for solving many nonconvex problems, particularly those of combinatorial type (see, e.g., [1, 2, 4, 10, 15]). In most applications in the nonconvex setting, the constraint sets satisfy some type of regularity property and local convergence can be proved [11, 18, 21]. Nonetheless, the results on global behavior are limited to very special sets [3, 12] and cannot explain the good performance of the algorithm in the nonconvex setting.

It is clear that the performance of an algorithm heavily depends on how the problem is formulated. When the Douglas–Rachford algorithm is applied to a nonconvex problem, finding an “adequate” formulation can be crucial for its success as a heuristic. Indeed, the formulation of the problem often determines whether or not the algorithm can successfully solve the problem at hand always, frequently or never.

Let us briefly present an interesting example exhibiting this behavior: Sudoku puzzles. Their solution with the Douglas–Rachford algorithm was first proposed in [15], and subsequently analyzed in [23] and [2]. A *Sudoku* is a  $9 \times 9$  grid, divided in nine  $3 \times 3$  subgrids, with some entries already prefilled. The objective is to fill the remaining entries in such a way that each row, column and subgrid contains the digits from 1 to 9 exactly once, see Figure 6.

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

(a) Unsolved Sudoku

4	2	6	5	7	1	3	9	8
8	5	7	2	9	3	1	4	6
1	3	9	4	6	8	2	7	5
9	7	1	3	8	5	6	2	4
5	4	3	7	2	6	8	1	9
6	8	2	1	4	9	7	5	3
7	9	4	6	3	2	5	8	1
2	6	5	8	1	4	9	3	7
3	1	8	9	5	7	4	6	2

(b) Solved Sudoku

Figure 6: Example of Sudoku

Then, Sudokus are nothing else than matrix completion problems, which can be modeled as feasibility problems. There are different possibilities for choosing

the constraint sets  $C_1, \dots, C_N$  in such a way that  $\bigcap_{i=1}^N C_i$  coincides with the (unique) solution to the Sudoku. Nonetheless, in order to apply the Douglas–Rachford algorithm, the sets must be chosen in such a way that the projections can be efficiently computed, ideally having a closed form. In [2, Section 6], two ways of modeling Sudokus are analyzed: as integer and binary feasibility problems. We briefly describe them in what follows.

### Sudoku modeled as an integer program

It is obvious how to model a Sudoku as an integer feasibility problem. Let  $S$  be the partially filled  $9 \times 9$  matrix representing the unsolved Sudoku, let  $J \subset \{1, 2, \dots, 9\}^2$  be the set of indices for which  $S$  is filled, and let  $A_{i,j}$  denote the  $(i, j)$ -th entry of the matrix  $A$ . If we denote by  $\mathcal{C}$  the set of vectors which are permutations of  $1, 2, \dots, 9$ , then a matrix  $A \in \mathbb{R}^{9 \times 9}$  is a solution to the Sudoku if and only if  $A \in C_1 \cap C_2 \cap C_3 \cap C_4$ , where

$$\begin{aligned} C_1 &:= \{A \in \mathbb{R}^{9 \times 9} : \text{each row of } A \text{ belongs to } \mathcal{C}\}, \\ C_2 &:= \{A \in \mathbb{R}^{9 \times 9} : \text{each column of } A \text{ belongs to } \mathcal{C}\}, \\ C_3 &:= \{A \in \mathbb{R}^{9 \times 9} : \text{each of the 9 subgrids of } A \text{ belongs to } \mathcal{C}\}, \\ C_4 &:= \{A \in \mathbb{R}^{9 \times 9} : A_{i,j} = S_{i,j} \text{ for each } (i, j) \in J\}. \end{aligned}$$

The projection onto  $C_4$  can be easily computed componentwise, while the projections onto  $C_1$ ,  $C_2$  and  $C_3$  are determined by the next result, see [2] for more details. Denote by  $\mathcal{C} \subset \mathbb{R}^m$  the set of vectors whose entries are all permutations of  $c_1, c_2, \dots, c_m \in \mathbb{R}$ . Then, for any  $x \in \mathbb{R}^m$ , one has

$$P_{\mathcal{C}}(x) = [\mathcal{C}]_x,$$

where  $[\mathcal{C}]_x$  denotes the set of vectors in  $\mathcal{C}$  such that  $y \in [\mathcal{C}]_x$  if the  $i$ th largest entry of  $y$  has the same index in  $y$  as the  $i$ th largest entry of  $x$ .

See [2, Proposition 5.1]. ■

### Sudoku modeled as a zero-one program

To model Sudoku puzzles as binary programs, we reformulate a matrix  $A \in \mathbb{R}^{9 \times 9}$  as  $B \in \mathbb{R}^{9 \times 9 \times 9}$ , where

$$B_{i,j,k} := \begin{cases} 1 & \text{if } A_{i,j} = k, \\ 0 & \text{otherwise.} \end{cases}$$

This reformulation transforms the entries into a 3-dimensional zero-one array, which can be thought as a cube, see Figure 7.

Let us denote by  $S'$  the  $9 \times 9 \times 9$  zero-one array corresponding to the unsolved Sudoku  $S$ , let  $I := \{1, 2, \dots, 9\}$ , let  $J' \subseteq I^3$  be the set of indices for which  $S'$  is filled, and let  $\mathcal{B}$  denote the 9-dimensional standard basis. The four constraints

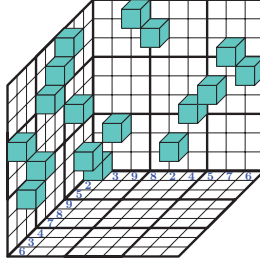


Figure 7: Zero-one representation of a Sudoku puzzle

of the integer program above become now

$$\begin{aligned}
 C_1 &:= \{B \in \mathbb{R}^{9 \times 9 \times 9} : B_{i,:k} \in \mathcal{B} \text{ for each } i, k \in I\}, \\
 C_2 &:= \{B \in \mathbb{R}^{9 \times 9 \times 9} : B_{:,j,k} \in \mathcal{B} \text{ for each } j, k \in I\}, \\
 C_3 &:= \{B \in \mathbb{R}^{9 \times 9 \times 9} : \text{vec } B_{3i+1:3(i+1), 3j+1:3(j+1), k} \in \mathcal{B} \text{ for } i, j = 0, 1, 2 \text{ and } k \in I\}, \\
 C_4 &:= \{B \in \mathbb{R}^{9 \times 9 \times 9} : B_{i,j,k} = 1 \text{ for each } (i, j, k) \in J'\},
 \end{aligned}$$

where  $\text{vec } A$  represents the *vectorization* of a matrix  $A$  by columns.

Further, since each cell in the Sudoku can only have one number assigned, we must add the additional constraint

$$C_5 = \{B \in \mathbb{R}^{9 \times 9 \times 9} : B_{i,j,:} \in \mathcal{B} \text{ for each } i, j \in I\}.$$

These constraints are represented in Figure 8.

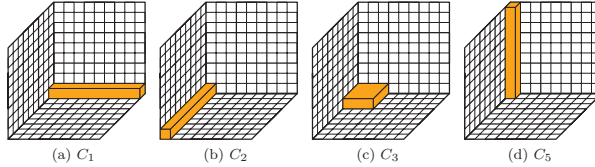


Figure 8: Visualization of the constraints used for modeling Sudoku as a zero-one program. The entries in the colored blocks are all 0 except for a single 1

Then,  $B$  completes  $S'$  (and thus solves the Sudoku) if and only if  $B \in C_1 \cap C_2 \cap C_3 \cap C_4 \cap C_5$ . Again, the projection onto  $C_4$  can be easily computed

componentwise, while the projections onto  $C_1$ ,  $C_2$ ,  $C_3$  and  $C_5$  are determined by Proposition 2.3.

### Performance of the Douglas--Rachford algorithm on Sudokus

As observed in [2], the Douglas--Rachford algorithm is totally ineffective for solving the integer formulation. On the other hand, the algorithm is very successful when it is applied to the binary formulation, being able to solve nearly all the instances (without a restart) in all the Sudoku libraries tested in [2]. Even in the one where it was most unsuccessful, the `top95`<sup>1</sup> library, it solved 87% of the instances.

Another possibility for modeling Sudoku as feasibility problems was tested in [4, 5], based on the fact that an unsolved Sudoku can be viewed as a graph precoloring problem. Surprisingly, the *rank formulation* proposed in [5] had a success rate of 100% in the `top95` library, and no Sudoku has been found so far for which Douglas--Rachford fails to find its solution for any starting point.

Therefore, it is clear that the choice of the formulation has a big impact in the success rate of the Douglas--Rachford algorithm when it is applied in the nonconvex setting.

### 3. Finding magic squares as a feasibility problem

In this next section, we propose two different formulations for finding magic squares as a feasibility problem which are inspired by the binary and integer formulations of Sudokus. To formulate the search, the first thing we need to do is to choose some sets whose intersection includes all the properties that characterize a magic square of order  $n$ : the sum of each row, column and main diagonals must be equal to the magic constant  $c := \frac{n(n^2+1)}{2}$ , and it must contain all the numbers between 1 and  $n^2$ . Note that if our purpose was to complete a prefilled magic square, we would need to add an additional constraint that fixes the prefilled numbers.

The most obvious way to look for a magic square is to use an integer formulation. This is explained in the next subsection, where we also deduce the expression for the projection onto each of the constraint sets. Afterwards, we show how to formulate the search for magic squares as a zero-one program, in line with what we have seen for Sudokus in Section 2.3, although in a slightly different way.

---

<sup>1</sup>`top95`: <http://magictour.free.fr/top95>

### 3.1. Magic squares modeled as an integer program

If we denote by  $\mathcal{P}$  the set of permutations of  $1, 2, \dots, n^2$  and  $I := \{1, 2, \dots, n\}$ , then  $A \in \mathbb{R}^{n \times n}$  is a magic square if and only if  $A \in \bigcap_{i=1}^5 C_i$ , where

$$\begin{aligned} C_1 &:= \left\{ A \in \mathbb{R}^{n \times n} : \sum_{j \in I} A_{i,j} = c \text{ for each } i \in I \right\}, \\ C_2 &:= \left\{ A \in \mathbb{R}^{n \times n} : \sum_{i \in I} A_{i,j} = c \text{ for each } j \in I \right\}, \\ C_3 &:= \left\{ A \in \mathbb{R}^{n \times n} : \sum_{i \in I} A_{i,i} = c \right\}, \\ C_4 &:= \left\{ A \in \mathbb{R}^{n \times n} : \sum_{i \in I} A_{i,n+1-i} = c \right\}, \\ C_5 &:= \{ A \in \mathbb{R}^{n \times n} : \text{vec } A \in \mathcal{P} \}. \end{aligned}$$

The first four sets are clearly convex, and their projection operators have a closed form determined by the next result. Consider  $S = \{x \in \mathbb{R}^m : \sum_{i=1}^m x_i = c\}$ . For any  $x \in \mathbb{R}^m$ ,

$$P_S(x) = x + \frac{1}{m} \left( c - \sum_{i=1}^m x_i \right) e, \quad \text{where } e = [1, 1, \dots, 1]^T.$$

This follows from the standard formula for the orthogonal projection onto a hyperplane, since  $S = \{x \in \mathbb{R}^m : \langle x, e \rangle = c\}$ ; see, e.g., [8, Example 3.23]. ■

Thereby, the projections onto each of these sets are given by

$$\begin{aligned} P_{C_1}(A) &= A + \frac{1}{n} \begin{pmatrix} c - \sum_{i=1}^n A_{1,i} & \cdots & c - \sum_{i=1}^n A_{1,i} \\ \vdots & & \vdots \\ c - \sum_{i=1}^n A_{n,i} & \cdots & c - \sum_{i=1}^n A_{n,i} \end{pmatrix}, \\ P_{C_2}(A) &= A + \frac{1}{n} \begin{pmatrix} c - \sum_{i=1}^n A_{i,1} & \cdots & c - \sum_{i=1}^n A_{i,n} \\ \vdots & & \vdots \\ c - \sum_{i=1}^n A_{i,1} & \cdots & c - \sum_{i=1}^n A_{i,n} \end{pmatrix}, \\ P_{C_3}(A) &= A + \frac{1}{n} \left( c - \sum_{i=1}^n A_{i,i} \right) I_n, \\ P_{C_4}(A) &= A + \frac{1}{n} \left( c - \sum_{i=1}^n A_{i,n+1-i} \right) \begin{pmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \ddots & 1 & 0 \\ 0 & \ddots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix}, \end{aligned}$$

where  $I_n$  denotes the identity matrix in  $\mathbb{R}^{n \times n}$ .

Finally,  $C_5$  is a nonconvex set containing all those matrices whose entries

are permutations of  $1, 2, \dots, n^2$ , so its projection operator is determined by Proposition 2.3. Therefore, given an  $n \times n$  matrix, to compute its projection onto  $C_5$ , we just need to sort the elements of the matrix in ascending order and place the number 1 in the cell that contains the smallest number, the number 2 in the cell that contains the next one, and so on. If there are two equal elements, the projection of the matrix is not unique (in our experiments in Section 4, we chose the element that appeared first when the matrix was read by rows).

Observe that another possibility here would be to combine all  $2n + 2$  sum constraints in the sets  $C_1, C_2, C_3$  and  $C_4$  into a single affine constraint set, whose projection operator can be explicitly computed (see, e.g., [8, Example 29.17]). The resulting model would have the additional advantage of not having to rely on the product space, so the iterates would belong to a space with smaller dimension.

### Completing a partially filled magic square

If  $M$  is a partially complete matrix representing an incomplete magic square, denote by  $J \subseteq I^2$  the set of indices for which  $M$  is filled. To find a completion of the magic square, we need to add the constraint

$$C_6 := \{A \in \mathbb{R}^{n \times n} : A_{i,j} = M_{i,j} \text{ for each } (i,j) \in J\}$$

to the set of constraints that we already have. Then,  $A$  completes  $M$  if, and only if,

$$A \in C_1 \cap C_2 \cap C_3 \cap C_4 \cap C_5 \cap C_6.$$

The projection onto  $C_6$  is given componentwise by

$$P_{C_6}(A_{i,j}) = \begin{cases} M_{i,j} & \text{if } (i,j) \in J, \\ A_{i,j} & \text{otherwise,} \end{cases}$$

for each  $(i,j) \in I^2$ .

### 3.2. Magic squares modeled as a binary program

To model the search for a magic square as a binary feasibility problem, we reformulate a matrix  $A \in \mathbb{R}^{n \times n}$  as  $B \in \mathbb{R}^{n \times n \times n^2}$ , where

$$B_{i,j,k} := \begin{cases} 1 & \text{if } A_{i,j} \geq k, \\ 0 & \text{otherwise.} \end{cases}$$

In this way, we transform the entries of the matrix into a 3-dimensional zero-one array, and each number in the magic square can be thought as a *pillar* made of small cubes, see Fig. 9.

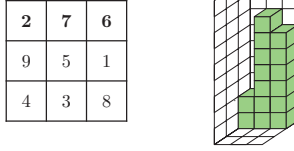


Figure 9: Representation of the numbers in the first row of a  $3 \times 3$  magic square as columns of small cubes for the binary formulation

The constraints from the previous subsection become (see 10):

$$\begin{aligned}
 C_1 &:= \left\{ B \in \mathbb{R}^{n \times n \times n^2} : \sum_{j=1}^n \sum_{k=1}^{n^2} B_{i,j,k} = c \text{ for each } i \in I \right\}, \\
 C_2 &:= \left\{ B \in \mathbb{R}^{n \times n \times n^2} : \sum_{i=1}^n \sum_{k=1}^{n^2} B_{i,j,k} = c \text{ for each } j \in I \right\}, \\
 C_3 &:= \left\{ B \in \mathbb{R}^{n \times n \times n^2} : \sum_{i=1}^n \sum_{k=1}^{n^2} B_{i,i,k} = c \right\}, \\
 C_4 &:= \left\{ B \in \mathbb{R}^{n \times n \times n^2} : \sum_{i=1}^n \sum_{k=1}^{n^2} B_{i,n+1-i,k} = c \right\}.
 \end{aligned}$$

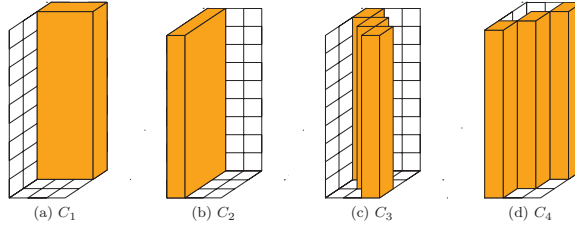


Figure 10: Visualization of the constraints used for modeling magic squares as a zero-one program. Each colored block must be formed by exactly  $c$  cubes (ones)

Constraint  $C_5$  of the integer formulation is now written for the binary for-

mulation as the intersection of two sets:

$$C_5 := \left\{ B \in \{0, 1\}^{n \times n \times n^2} : \sum_{i=1}^n \sum_{j=1}^n B_{i,j,k} = n^2 - k + 1, \forall k = 1, \dots, n^2 \right\},$$

$$C_6 := \left\{ B \in \{0, 1\}^{n \times n \times n^2} : \text{vec } B_{i,j,:} \in \mathcal{B} \text{ for each } i, j \in I \right\},$$

where  $\mathcal{B} := \{[1, 1, \dots, 1, 1], [1, 1, \dots, 1, 0], \dots, [1, 0, \dots, 0, 0]\}$  is a base of vectors of  $\mathbb{R}^{n^2}$ . On the one hand, constraint  $C_5$  guarantees that the first *floor* of  $B$  is all filled with ones, the second must have  $n^2 - 1$  ones, and so on until we get to the top *floor*, which only contains a one. On the other hand, constraint  $C_6$  guarantees that the matrix is formed by *pillars* of ones *standing* on the *floor*, so if there is a one in an entry, all the elements below must be ones as well. Then,  $B \in \mathbb{R}^{n \times n \times n^2}$  is a magic square if and only if  $B \in \bigcap_{i=1}^6 C_i$ .

The projections onto the first four sets are given by 3.1:

$$P_{C_1}(B) = B + \frac{1}{n^3} \begin{pmatrix} u_1 & \cdots & u_1 \\ u_2 & \cdots & u_2 \\ \vdots & & \vdots \\ u_n & \cdots & u_n \end{pmatrix},$$

$$P_{C_2}(B) = B + \frac{1}{n^3} \begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ \vdots & \vdots & & \vdots \\ v_1 & v_2 & \cdots & v_n \end{pmatrix},$$

$$P_{C_3}(B) = B + \frac{1}{n^3} \left( c - \sum_{i=1}^n \sum_{k=1}^{n^2} B_{iik} \right) \begin{pmatrix} e & 0_{n^2} & \cdots & 0_{n^2} \\ 0_{n^2} & e & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0_{n^2} \\ 0_{n^2} & \cdots & 0_{n^2} & e \end{pmatrix},$$

$$P_{C_4}(B) = B + \frac{1}{n^3} \left( c - \sum_{i=1}^n \sum_{k=1}^{n^2} B_{i(n+i-1)k} \right) \begin{pmatrix} 0_{n^2} & \cdots & 0_{n^2} & e \\ \vdots & \ddots & e & 0_{n^2} \\ 0_{n^2} & \ddots & \ddots & \vdots \\ e & 0_{n^2} & \cdots & 0_{n^2} \end{pmatrix}.$$

where  $e = (1, \dots, 1) \in \mathbb{R}^{n^2}$  and

$$u_p := \left( c - \sum_{j=1}^n \sum_{k=1}^{n^2} B_{p,j,k}, \dots, c - \sum_{j=1}^n \sum_{k=1}^{n^2} B_{p,j,k} \right) \in \mathbb{R}^{n^2},$$

$$v_p := \left( c - \sum_{i=1}^n \sum_{k=1}^{n^2} B_{i,p,k}, \dots, c - \sum_{i=1}^n \sum_{k=1}^{n^2} B_{i,p,k} \right) \in \mathbb{R}^{n^2}.$$

The projection onto  $C_5$  assigns a one to all the elements of the first *floor*, a zero to the smallest element and a one to the rest in the second *floor*, a zero



to the two smallest elements and a one to the rest in the third *floor*, and so on until the top *floor*, which only contains a one. The projection onto  $C_6$  can be computed componentwise as follows: For each  $i, j \in I$ , we have

$$P_{C_6}(B)_{i,j,:} = \operatorname{argmin}_{b \in \mathcal{B}} \|B_{i,j,:} - b\|.$$

The formulation of the problem of completing a partially filled magic square is analogous to the integer formulation, so we omit the details.

#### 4. Numerical experiments

In this section we run a numerical experiment to test the performance of the Douglas--Rachford algorithm for finding magic squares of order  $n = 3, 4, \dots, 12$ , on which we compare the integer and binary formulations discussed in Section 3. All our codes were written in Python 2.7 and the tests were run on an Intel Core i7-4770 CPU 3.40GHz with 32GB RAM, under Windows 10 (64-bit).

For each order and each formulation, the algorithm was run on the corresponding product space (see Section 2.2) for 100 random starting points. For the integer formulation, we took a starting point of the form  $x_0 := (y, y, y, y, y) \in D$  for some  $y \in \mathbb{R}^{n \times n}$  randomly chosen with entries in  $]0, 1[$ , while for the binary formulation, we choose  $x_0 := (y, y, y, y, y) \in D$  for some random  $y \in \mathbb{R}^{n \times n \times n^2}$  with entries in  $]0, 1[$ . The iterates were then defined by

$$x_{n+1} = T_{D,C}(x_n) \quad \text{for } n = 0, 1, \dots$$

The Douglas--Rachford algorithm was stopped either after a maximum time of 1800 seconds (30 minutes) or when  $\operatorname{round}(P_D(x_n))$  satisfies all the constraints, where  $\operatorname{round}(\cdot)$  gives the nearest integer componentwise.

Table 1 summarizes the success rate and running time for each formulation. We observe how the binary formulation was not able to solve any instance for orders larger than 6. On the other hand, for orders 4 and 5, it was more successful than the integer formulation, which surprisingly becomes more successful for larger orders than for smaller ones. The algorithm was much faster with the integer formulation.

To better understand the behavior of the Douglas--Rachford algorithm for both formulations, we show in Figure 11 the cumulative frequency over time. Note that we have used a logarithmic scale in the horizontal axis, to show how the running time of the algorithm exponentially grows with the order  $n$ . Apparently, the reason why the algorithm failed in the binary formulation for  $n$  larger than 6 is that it did not have enough time to converge. We show in Figure 12 a magic square of orders between 3 and 12 found with the integer formulation.

Order	Binary			Integer		
	Success	Time		Success	Time	
3	99	0.05	(0.17)	100	0.01	(0.02)
4	93	1.67	(13.02)	64	0.06	(2.64)
5	79	436.86	(1739.93)	59	0.67	(5.63)
6	3	1228.78	(1742.74)	80	0.53	(2.55)
7	0	-	-	86	0.74	(4.92)
8	0	-	-	94	0.86	(4.39)
9	0	-	-	96	0.90	(3.39)
10	0	-	-	94	1.47	(8.69)
11	0	-	-	97	2.45	(13.97)
12	0	-	-	99	4.86	(22.97)
13	0	-	-	98	9.64	(56.71)
14	0	-	-	100	24.31	(84.90)
15	0	-	-	100	60.04	(236.27)
16	0	-	-	100	245.58	(1291.61)

Table 1: Magic squares found in less than 30 minutes and mean (max) time

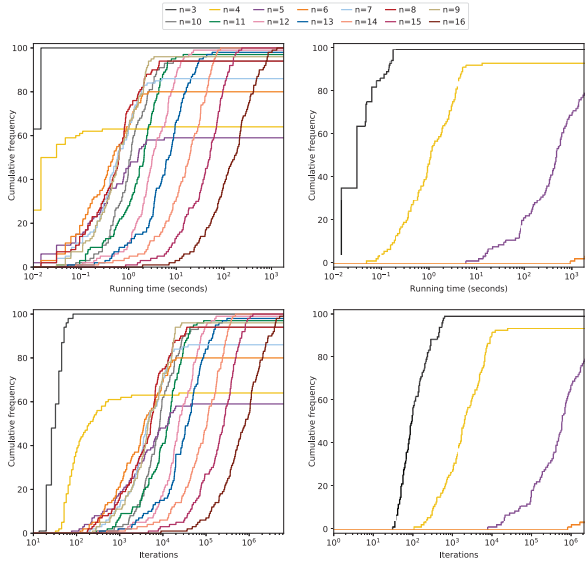


Figure 11: Cumulative frequencies over running time of the integer (left) and binary (right) formulations

8	3	4	11	14	5	4	2	14	20	11	18	6	35	2	31	29	8	13	4	33	28	38	18	41		
1	5	9	16	2	9	7	19	9	5	8	24	5	27	17	9	19	34	10	22	30	34	25	17	37		
6	7	2	1	15	8	10	15	17	23	3	7	14	13	23	15	16	30	43	44	15	29	14	6	24		
			6	3	12	13	25	12	1	21	6	32	10	26	12	28	3	16	47	1	31	5	48	27		
							4	13	16	22	10	33	22	7	20	18	11	8	26	21	23	49	45	3		
												21	4	36	24	1	25	46	12	40	19	42	9	7		
																		39	20	35	11	2	32	36		
10	15	41	58	54	23	34	25	71	14	40	68	28	19	22	70	37	76	64	57	12	17	59	97	33	74	7
33	55	3	28	49	47	38	7	46	29	59	21	58	62	51	10	33	69	49	34	39	75	9	40	52	48	90
14	50	32	1	59	57	42	5	43	64	1	66	47	24	78	16	30	66	24	22	72	100	78	81	6	2	54
51	40	27	18	11	43	31	39	43	73	36	5	34	77	52	50	38	19	37	68	84	14	18	92	55	91	27
20	30	61	63	17	4	36	29	3	20	67	56	65	26	17	55	60	8	73	10	13	12	89	96	70	51	83
44	37	22	13	12	62	6	64	45	49	69	11	12	75	44	25	39	58	31	62	86	87	23	32	95	5	26
53	9	48	19	56	8	21	46	74	80	2	57	35	31	9	18	63	61	79	38	60	35	93	16	11	71	41
35	24	26	60	2	16	52	45	7	13	54	79	42	32	81	53	8	67	46	43	50	25	44	1	99	45	85
							76	27	41	6	48	23	15	72	61		28	20	94	65	42	29	3	80	88	56
																53	82	77	15	98	63	47	4	30	36	
114	61	60	36	88	2	78	73	52	72	35	19	131	79	107	103	58	59	35	16	109	60	94				
119	46	81	117	98	6	42	26	20	91	25	104	25	61	38	87	138	37	140	124	39	31	46				
93	7	40	104	109	17	77	31	79	69	45	67	30	64	137	49	69	17	101	72	34	113	117				
3	71	51	32	83	80	84	15	59	75	118	76	126	55	132	13	23	84	73	57	82	129	20				
62	105	37	21	54	113	102	48	9	10	110	50	108	134	68	78	71	9	97	41	24	88	102				
28	11	64	38	47	87	65	97	89	29	116	51	112	22	7	114	139	128	10	75	111	47	54				
44	16	115	120	24	94	18	106	14	50	70	8	123	4	89	48	127	26	70	121	40	98	116				
99	112	23	12	5	101	86	43	49	74	67	86	5	28	2	91	100	125	136	66	106	80	45				
34	68	96	33	13	30	92	55	103	90	57	133	12	141	11	93	3	130	144	18	56	77	52				
53	108	41	100	39	56	8	82	76	107	1	143	43	120	83	99	33	44	21	110	85	36	53				
22	66	63	58	111	85	19	95	121	4	27	118	65	27	113	63	95	92	42	96	122	6	29				
											15	90	135	81	32	14	119	1	74	62	105	142				

Figure 12: Magic squares of orders 3 to 12 found with the integer formulation

## 5. Conclusions

In this article, we have shown that the Douglas--Rachford algorithm can be used as a successful heuristic for constructing magic squares, despite the nonconvexity of the problem. To this aim, we have proposed two different formulations as a feasibility problem, one integer and another one binary. Our numerical tests demonstrate that the integer formulation is more effective for finding magic squares because it requires considerably much less time on average for finding a solution. Surprisingly, this behavior was the opposite with the similar binary and integer formulations for solving Sudoku puzzles, where the integer formulation is completely unsuccessful. Interestingly, the binary was more successful than the integer formulation for orders 4 and 5. For higher orders, the binary formulation was not able to solve any instance, possibly due to a lack of time.

## Acknowledgements

We would like to thank the referees for their careful reading and comments, and to Veit Elser for his useful suggestions. The first author was supported by MINECO of Spain and ERDF of EU, as part of the Ramón y Cajal program (RYC-2013-13327) and Grants MTM2014-59179-C2-1-P and PGC2018-097960-B-C22.

## References

- [1] Aragón Artacho, F.J., Borwein, J.M. and Tam, M.K. (2014). Douglas--Rachford feasibility methods for matrix completion problems. *ANZIAM J.* **55**, 299--326.
- [2] Aragón Artacho, F.J., Borwein, J.M. and Tam, M.K. (2014). Recent results on Douglas--Rachford methods for combinatorial optimization problems. *J. Optim. Theory Appl.* **163**, 1--30.
- [3] Aragón Artacho, F.J., Borwein, J.M. and Tam, M.K. (2016). Global behavior of the Douglas--Rachford method for a nonconvex feasibility problem. *Aceptado en J. Global Optim.* **65**, 309--327.
- [4] Aragón Artacho, F.J. and Campoy, R. (2018). Solving graph coloring problems with the Douglas--Rachford algorithm. *Set-Valued Var. Anal.* **26**(2), 277--304.
- [5] Aragón Artacho, F.J., Campoy, R. and Elser, V. (2018). An enhanced formulation for solving graph coloring problems with the Douglas--Rachford algorithm. ArXiv: 1808.01022

- [6] Aragón Artacho, F.J., Censor, Y. and Gibali, A. (2018). The cyclic Douglas--Rachford algorithm with  $r$ -sets-Douglas--Rachford operators. *Optimization Methods and Software*, 1–15. DOI: 10.1080/10556788.2018.1504049
- [7] Baillon, J.B., Bruck, R.E. and Reich, S. (1978). On the asymptotic behavior of nonexpansive mappings and semigroups in Banach spaces. *Houston J. Math.* **4**(1), 1–9.
- [8] Bauschke, H.H. and Combettes, P.L. (2017). *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, 2nd edition, Springer, New York.
- [9] Bauschke, H.H., Combettes, P.L. and Luke, D.R. (2004). Finding best approximation pairs relative to two closed convex sets in Hilbert space. *J. Approx. Theory*. **127**, 178–192.
- [10] Bauschke, H.H. and Koch, V.R. (2015). Projection methods: Swiss army knives for solving feasibility and best approximation problems with half-spaces. *Contemp. Math.* **636**, 1–40.
- [11] Bauschke, H.H. and Noll, D. (2014). On the local convergence of the Douglas--Rachford algorithm. *Arch. Math.* **102**(6), 589–600.
- [12] Benoist, J. (2015). The Douglas--Rachford algorithm for the case of the sphere and the line. *J. Global Optim.* **63**(2), 363–380.
- [13] Borwein, J.M. and Tam, M.K. (2014). A cyclic Douglas--Rachford iteration scheme. *J. Optim. Theory Appl.* **160**(1), 1–29.
- [14] Douglas, J. and Rachford, H.H. (1956). On the numerical solution of heat conduction problems in two and three space variables. *Trans. Amer. Math. Soc.* **82**, 421–439.
- [15] Elser, V., Rankenburg, I. and Thibault, P. (2007). Searching with iterated maps. *Proc. Natl. Acad. Sci.* **104**(2), 418–423.
- [16] Escalante, R. and Raydan, M. (2011). *Alternating Projection Methods*. Fundamentals of Algorithms **8**, Society for Industrial and Applied Mathematics (SIAM), Philadelphia.
- [17] Goebel, K. and Kirk, W.A. (1990). *Topics in Metric Fixed Point Theory*, Cambridge University Press, Cambridge.
- [18] Hesse, R. and Luke, D.R. (2013). Nonconvex notions of regularity and convergence of fundamental algorithms for feasibility problems. *SIAM J. Optim.* **23**(4), 2397–2419.

- [19] Lions, P.L. and Mercier, B. (1979). Splitting algorithms for the sum of two nonlinear operators. *SIAM J. Numer. Anal.* **16**(6), 964–979.
- [20] Opial, Z. (1967). Weak convergence of the sequence of successive approximations for nonexpansive mappings. *Bull. Amer. Math. Soc.* **73**(4), 591–597.
- [21] Phan, H.M. (2016). Linear convergence of the Douglas–Rachford method for two closed sets. *Optim.* **65**(2), 369–385.
- [22] Pierra, G. (1984). Decomposition through formalization in a product space. *Math. Program.* **28**(2), 96–115.
- [23] Schaad, J. (2010). *Modelling the 8-queens problem and Sudoku using an algorithm based on projections onto nonconvex sets*. Masters’s thesis, Univ. British Columbia (Canada).
- [24] Segura Martínez, P. (2017). *Resolución de cuadrados mágicos con el algoritmo de Douglas–Rachford*. Final year undergraduate project, Univ. Alicante (Spain). <http://hdl.handle.net/10045/67558>.
- [25] Svaiter, B.F. (2011). On weak convergence of the Douglas–Rachford method. *SIAM J. Control Optim.* **49**(1), 280–287.
- [26] Swetz, F.J. (2008). *Legacy of the Luoshu: The 4,000 Year Search for the Meaning of the Magic Square of Order Three*, A K Peters, Ltd., Wellesley, Massachusetts.
- [27] Taylor, G. (2003). A secret message in Dürer’s magic square? *Notes in the History of Art.* **23**(1), 17–22.

#### About the authors

**Francisco J. Aragón Artacho** earned his Ph.D. in 2007 from the University of Murcia (Spain). He subsequently held a postdoctoral position “Juan de la Cierva” at the University of Alicante. In 2011 he moved to the University of Newcastle (Australia), where he became a Research Associate at the Research Centre for Computer-Assisted Research Mathematics and its Applications. Two years later, he moved to Luxembourg where he became a Marie Curie Fellow (AFR Postdoc) at the Luxembourg Centre for Systems Biomedicine. In November 2014 he started his current position at the University of Alicante as a Ramón y Cajal fellow. His main research interests include set-valued and variational analysis, optimization, convex analysis and projection algorithms.

**Paula Segura Martínez** graduated from the University of Alicante with a B.S. in Mathematics in 2017. She received her M.S. in Mathematics from the University of Murcia in 2018 and now she is a Ph.D. student at the University of

Valencia. Her research interests are in the fields of combinatorial optimization, location and arc routing problems.